ASSEMBLY CODE

```
ORG 0000H
MOV B,#0AH          //storing divisor value in B
MOV A,#93H          //storing dividend value in A
MOV 20H,A           //storing A value in internal memory
MOV 21H,A           //storing A value in internal memory
MOV 22H,B           //storing B value in internal memory
MOV A,B             //storing B value in A
JZ ERROR_DIVISOR_ZERO         //checking if the value in A is ZERO if it is,
jump to error_divisor_zero
RLC A               //left shifting value in A(multiply by 2)
JC ERROR_8BIT_EXCEED          //checking if the value is overflowing if it
is, jumping to exceed_8bit_exceed
RLC A               //left shifting value in A(multiply by 2)
JC ERROR_8BIT_EXCEED          //checking if the value is overflowing if it
is, jumping to exceed_8bit_exceed
CLR C               //clearing carry flag
MOV B,A             //storing divisor value in B
MOV R0,#0H          //Initiating R0 register to zero to store quotient
MOV A,20H           //storing dividend value in A

DIVISIONLOOP:       //loop to divide
MOV R1,A            //storing A value in R1 for future reference
SUBB A,B            //subtracting B from A
JC DIVISIONEND      //Jump to DIVISIONEND if A becomes negative
INC R0                    //Incrementing R0 to store quotient
SJMP DIVISIONLOOP //Jump to DIVISIONLOOP to continue the division

DIVISIONEND:        //Loop to end the division

MOV 24H,R0          //Storing R0 in memory address 24H
MOV 25H,R1          //Storing R1 in memory address 25H
MOV 30H,#00H        //Storing 0 in memory address 30H
SJMP ENDLOOP        //Jump to ENDLOOP


ERROR_DIVISOR_ZERO:           //error when the divisor is zero
MOV 30H,#01H        // Storing 1 in memory address 30H
SJMP ENDLOOP        // Jump to ENDLOOP

ERROR_8BIT_EXCEED:            //error when bits overflows
MOV 30H,#02H        // Storing 2 in memory address 30H
SJMP ENDLOOP        // Jump to ENDLOOP

ENDLOOP:
SJMP ENDLOOP        // Jump to ENDLOOP, creates an infinite loop

END
```